User Guide
DLN:Inspector
Version 1.0 Beta
License

User Guide DLN:Inspector, version 1.0 Beta

© 2018 Cultural Heritage Imaging

This work is licensed under the Creative Commons Attribution-Noncommercial-No Derivative Works 3.0 United States License. To view a copy of this license, visit http://creativecommons.org/licenses/by-nc-nd/3.0/us/ or send a letter to Creative Commons, 171 Second Street, Suite 300, San Francisco California 94105 USA.

Attribution should be made to Cultural Heritage Imaging, http://CulturalHeritageImaging.org.

The DLN:Inspector version 1.0 Beta software described in this guide is ©Cultural Heritage Imaging 2014 – 2018 and is made available under the Gnu General Public License version 3.

Acknowledgments

With the Beta release of the first two software tools in the Digital Lab Notebook (DLN) suite, CHI gratefully acknowledges the contributed expertise and funding that have made this work possible to date.

Software Development

DLN:CaptureContext: This tool was developed by Carla Schroer and Mark Mudge of CHI, along with Martin Doerr and Erich Leisch from the Foundation for Research and Technology Hellas (FORTH). Martin Doerr led the development of the semantic knowledge management structure. Erich Leisch wrote the DLN:CaptureContext software. All members of the team contributed to the design of the software requirements and specifications documents.

DLN:Inspector: Mark Mudge and Carla Schroer of CHI and senior software engineer Ron Bourret designed the Inspector tool in coordination with Martin Doerr and Erich Leisch of FORTH. Ron Bourret wrote the DLN:Inspector software.

Funding

Finalizing the 1.0 Beta release for the DLN tools is funded by a current and ongoing National Endowment for the Humanities Preservation and Access Research and Development grant: The Democratization of Scientific Imaging through Metadata Management and Archival Submission Support. The Centre for Cultural Informatics of ICS FORTH is CHI’s partner in this grant. The grant period is from January 2018 – June 2020. (Award # PR-258746-18)

A grant from the National Park Service’s National Center for Preservation Technology and Training (NCPTT), Applying Scientific Rigor to Photogrammetric 3D Documentation for Cultural Heritage and Natural Science Materials. This grant paid for the addition of photogrammetry support to the tools. This project was completed in December 2017.
A Digital Humanities start-up grant from the National Endowment for the Humanities, *Data Sustainability and Advanced Metadata Management for Scientific Imaging*. This funding paid for development of the DLN:Inspector tool, refinements to the DLN:CaptureContext tool, and an initial case study. This project was completed in December 2015. (Award # HD-51978-14)

Funds from earlier grants were used for initial investigations into approaches for process metadata in computational photography. This work included reviews of metadata strategies and use of the CIDOC Conceptual Reference Model, as well as early prototypes of the DLN:CaptureContext tool:

A Cyber-enabled Discover and Innovation grant from the National Science Foundation (NSF), *Automated Documentation and Illustration of Material Culture through the Collaborative Algorithmic Rendering Engine (CARE)*, in partnership with Princeton University. Completed in September 2014. (Award # IIS-1027962)

A National Leadership Grant from the Institute of Museum and Library Services, *Developing Advanced Technologies for the Imaging of Cultural Heritage Objects*, in partnership with University of Southern California. This project was completed in December 2009. (Award # LG-25-06-0107-06)

Additional funding came from public contributions to Cultural Heritage Imaging.

**Collaborators**

Over the history of Digital Lab Notebook development, many CHI collaborators, too many to name here, have contributed valuable advice and insights to the project. CHI is particularly grateful to these key collaborators:

Steve Stead, Tom Malzbender, Judy Bogart, Dominic Oldman, Adam Rabinowitz, Neffra Matthews, Tom Noble, Julian Richards, Kieron Niven, Chris Edwards, Dale Kronkright, Mary Elings, Stuart Snyderman, Susan Kane, Mohammed Elfarargy, Andrew G. Vaughn, Diane Manning

**Additional Resources**

Download the software and User Guide from CulturalHeritageImaging.org/downloads

See demonstrations of the software as part of the instructional video series “Simplifying Scientific Imaging” here: https://vimeo.com/channels/digitallabnotebook

Ask questions and take part in the conversation about these tools on the Free CHIForums: http://forums.culturalheritageimaging.org
Table of Contents

1. Introduction
2. Installing and starting the DLN:Inspector
   2.1. Installing the DLN:Inspector
   2.2. Updating the DLN database
   2.3. Starting the DLN:Inspector the first time
3. Using the DLN:Inspector
   3.1. Understanding projects, groups, and image sets
   3.2. Inspecting images
       3.2.1. DLN:Inspector for RTI, without the DLN database
       3.2.2. DLN:Inspector for RTI, with the DLN database
       3.2.3. DLN:Inspector for Photogrammetry, without the DLN database (browsing)
       3.2.4. DLN:Inspector for Photogrammetry, without the DLN database (custom)
       3.2.5. DLN:Inspector for Photogrammetry, with the DLN database
   3.3. Inspection results
       3.3.1. Saving inspection results
       3.3.2. Loading inspection results
       3.3.3. Understanding inspection results
   3.4. Setting options
Appendix A: Installation and startup
   A.1. Updating the DLN database
       A.1.1. On MacOS
       A.1.2. On Windows
   A.2. Solving startup problems
       A.2.1. Database installed but not running (or other connection problem)
       A.2.2. Database installed and running but not updated
   A.3. Creating custom directory patterns (advanced)
Appendix B: Results
Appendix C: Properties
   C.1. Property files
   C.2. Property prefixes
   C.3. International standards
1 Introduction

The DLN:Inspector checks the consistency of metadata across the image files used to generate Reflectance Transformation Imaging (RTI) files and photogrammetry models.

Metadata is data about data. In this case, it is data about images, such as the make of the camera used to create the image, the f-stop used to capture the image, and any transformations performed on the image, such as changes to brightness. If the images used to generate an RTI or photogrammetry model are created with different processes — for example, they use different f-stops — the resulting model will be less accurate.

The results generated by the DLN:Inspector are stored in the Digital Lab Notebook (DLN), if one is present. The DLN (a PostgreSQL database) is just a digital version of the paper notebooks used by scientists before the digital age and is created when you install the DLN Capture Context (DLNCC) tool. To get the DLNCC tool and PostgreSQL, please contact CHI. For more information, see http://culturalheritageimaging.org/Technologies/Digital_Lab_Notebook/.

In summary:

• The DLN:Inspector helps you improve your digital representations by checking that the metadata meets requirements for the technology you are applying.

• The DLN:Inspector has different versions for RTI and photogrammetry, as each checks different metadata.

• The DLN:Inspector is designed to run both with and without the DLN database.

2 Installing and starting the DLN:Inspector

The DLN:Inspector is available in separate versions for Windows and Mac OS.

2.1 Installing the DLN:Inspector

To install the DLN:Inspector:

• On Mac, copy the Inspector-PG and Inspector-RTI applications in the disk image into the /Applications folder.

• On Windows, run the installer.

2.2 Updating the DLN database

If you have installed DLNCC tool, see section A.1, “Updating the DLN database.” This explains how to update the DLN database for use with the DLN:Inspector.

If you have not installed the DLNCC tool or you do not want the DLN:Inspector to use the DLN database, you can skip this step.
2.3 Starting the DLN:Inspector the first time

When you run the DLN:Inspector for the first time, it tries to connect to the database. If you have not installed the database:

1. The DLN:Inspector displays an error message and asks if you want to continue without the database. Click “Yes.”

2. The DLN:Inspector asks if you want to use the database in the future. Click “No.” (You can change your answer later. For more information, see section 3.3, “Setting Options.”)

The DLN:Inspector continues without the database.

If you have installed and updated the database and it is running, the DLN:Inspector starts normally. If your situation is different — for example, you have installed the database but not updated it, or the database is not running — the DLN:Inspector asks a series of questions. For more information, see appendix A.2, “Solving startup problems.”

3 Using the DLN:Inspector

There are two versions of the DLN:Inspector — one to inspect images used to create RTI files and one to inspect images used to create photogrammetry models. Each version is designed to run with or without the DLN database.

To run the DLN:Inspector, click or double-click on the DLN:Inspector’s icon, as appropriate.

3.1 Understanding projects, groups, and image sets

An image set is a set of processed images used to create an RTI file or a photogrammetry model, and an optional set of archived images from which the processed images were created. For RTI, the processed images are always JPEG files; for photogrammetry, they can be JPEG or TIFF files. The archived images are always DNG files.

For RTI, the DLN:Inspector expects image sets to be gathered into projects, with a project directory containing related image set directories. In turn, each image set directory is expected to have a jpeg-exports subdirectory, which contains JPEG files, and an optional original-captures subdirectory, which contains DNG files. For example:

    Museum of History Project
    Bowl
        jpeg-exports
        original-captures
    Cup
        jpeg-exports
        original-captures
    ...

For photogrammetry, the DLN:Inspector expects one or more related image sets to be gathered into a group. The DLN:Inspector is quite flexible in how image sets and files are stored within a group directory. As shipped, it supports several common directory structures. For example, the following structure has a group directory containing image set directories. These contain image type directories, which contain images:

```
Cup
  Image set 1
  DNG files
  JPEG files
  Image set 2
  DNG files
  TIFF files
```

Advanced users can configure the DLN:Inspector to use custom directory structures. Any user can define the files in an image set by choosing them from anywhere on disk.

### 3.2 Inspecting images

To inspect images:

1. Find or define your image sets. How you do this depends on whether you are running the DLN:Inspector for RTI or Photogrammetry, and whether you are using the DLN database.

2. Select the image sets to inspect.

3. Click the “Inspect” button.

The following sections describe how to find or define image sets.
3.2.1 DLN:Inspector for RTI, without the DLN database

The DLN:Inspector displays the following dialog:

To find your image sets:

1. Click “Browse” and select the project directory.

2. The DLN:Inspector displays the names of project subdirectories that have jpeg-exports subdirectories. For example, if your directory structure is the following:

   Museum of History Project
   Cup
   original-captures
   jpeg-exports
   Bowl
   original-captures
   jpeg-exports

   the DLN:Inspector displays the names Cup and Bowl.

If you use a different directory structure (uncommon), see appendix A.3, “Creating custom directory patterns (advanced).”
### 3.2.2 DLN:Inspector for RTI, with the DLN database

The DLN:Inspector displays the following dialog:

![DLN:Inspector for RTI](image)

To find your image sets:

1. Select the project. The DLN:Inspector retrieves the groups in the project.
2. Select the group. The DLN:Inspector displays the image sets in the group.
3.2.3 DLN:Inspector for Photogrammetry, without the DLN database (browsing)

The DLN:Inspector displays the following dialog:

![DLN:Inspector for Photogrammetry](image)

The DLN:Inspector lets you choose from three common directory patterns. In each pattern, the term Group refers to the group directory, Image-set refers to a directory whose name is used as the image set name, Type-dir refers to a directory that contains an image type name, and Files refers to the actual files. For a type directory to “contain” an image type name, it must contain the words “JPEG,” “JPG,” “TIFF,” “TIF,” or “DNG” in any case. Examples are tifs, MyJPEGFiles, and DNG images.

To see an example of a directory pattern, click the example link. For example, clicking “example” for the first pattern displays the following dialog:
To find your image sets:

1. Select a directory pattern.

2. Click “Browse” and select the group directory.

3. The DLN:Inspector displays the names of the directories that match the “Image-set” level in the directory pattern you chose. For the example pattern shown above, this would be Image set 1 and Image set 2.

If your directory structure does not match any of these patterns, see section 3.2.4, “DLN:Inspector for Photogrammetry, without the DLN database (custom),” or appendix A.3, “Creating custom directory patterns (advanced).”

3.2.4 DLN:Inspector for Photogrammetry, without the DLN database (custom)

If your directory structure does not match one of the patterns shipped with the DLN:Inspector, you have two choices:

- **Define custom image sets.** This is a good choice if one type of images reside in multiple directories, processed and archived images reside in the same directory, or you want an easy-to-use solution. Continue in this section.

- **Define custom directory patterns.** This is a good choice if you have a regular directory structure that is different from the predefined structures, you are careful, and you are willing to learn. See appendix A.3, “Creating custom directory patterns (advanced).”
To define custom image sets:

1. Click on the Custom drop-down in the upper right corner of the DLN:Inspector. The DLN:Inspector displays the following dialog:

   ![Custom Drop-Down Dialog](image)

   Image sets:
   - Add...
   - Edit...
   - Remove
   - Save...
   - Load...

2. Click “Add”. The DLN:Inspector displays the following dialog:

   ![Define Image Set Dialog](image)

   Define Image Set
   - Name:
   - File type:
     - JPEG
     - TIFF
   - JPEG Files
   - DNG Files
   - Add...
   - Remove
   - OK
   - Cancel
3. Enter the image set name.

4. Select JPEG or TIFF as the type of your processed images and use “Add” to select the processed files in the image set.

5. If you have archived (DNG) images, use “Add” to select the archived files in the image set.

6. Click “OK” when you are done.

After you have returned to the previous dialog, you will probably want to save your definitions. This will prevent you from having to reenter your definitions next time you run the DLN:Inspector. This is especially important if you have many images in your image set.

To save your image set definitions:

1. Select one or more image sets whose definitions you want to save. If you don’t select any image sets, the definitions for all image sets will be saved.

2. Click “Save.”

You can also click the following buttons:

- **Add.** Add more image set definitions.
- **Edit.** Edit the selected image set definition.
- **Remove.** Remove the selected image set definition(s).
- **Load.** Load image set definitions from a file. Note that this discards the current image set definitions.
3.2.5 DLN:Inspector for Photogrammetry, with the DLN database

The DLN:Inspector displays the following dialog:

To find your image sets:

1. Select the project. The DLN:Inspector retrieves the groups from the database.
2. Select the group. The DLN:Inspector displays the image sets in the group.
3. By default, the DLN:Inspector uses all of the images in the image set directories you entered with the DLNCC tool. If you want to use a different set of files, click “Edit.” There are several common reasons for doing this:

   • Your directories contain images for multiple image sets. For example, you have 100 processed images and you want to define image sets for images 1–50, 25–75, and 50–100.
   • Your directories contain images that are not part of the image set that you want to omit. For example, you have a color card image in your archived images directory.
   • You did not enter a image set directories with the DLNCC tool and you need to add images to the image set definition.
The DLN:Inspector displays the following dialog.

![Define Image Set Dialog](image)

4. Select JPEG or TIFF as the type of your processed images and use “Add” and “Remove” to select the processed files in the image set. If you specified a processed images directory in the DLNCC tool, your images must be in that directory. If you did not specify a processed images directory, your images can be in any directory.

5. If you have archived (DNG) images, use “Add” and “Remove” to select the archived files in the image set. If you specified an archived images directory in the DLNCC tool, your images must be in that directory. If you did not specify an archived images directory, your images can be in any directory.

6. Click “OK.” The DLN:Inspector automatically saves your image set definition to the database.
3.3 Inspection results

After you click “Inspect,”, inspection results are displayed in the following window:

![Results Window]

This columns in this window are:

- **Image set.** The image set to which the result applies. This is necessary when you inspect more than one image set at a time.

- **Result.** One of the following:
  - **OK.** The images passed the test.
  - **Warning.** A non-fatal error occurred. For example, a value was outside its expected range.
  - **File error.** There was a problem with the set of image files. For example, DNG files were present for some, but not all, JPEG files.
  - **Capture error.** An error occurred while capturing images. For example, different images were taken with different lenses.
  - **Transformation error.** An illegal transformation was performed (for example, the brightness was changed) or a legal transformation was performed differently for different images (for example, two images have different color temperatures).
  - **Fatal error.** An error occurred in the DLN:Inspector. These are generally configuration errors.

For more information, see Appendix B, “Results”.
• **Property.** The name of the property to which the result applies. For more information, see Appendix C, “Properties.”

• **Description.** A short description of the result. Click on this result for more details.

### 3.3.1 Saving inspection results

When you are using the DLN database, the DLN:Inspector automatically saves results to the database. Regardless of database use, you can save results to a text file or an HTML file:

- To save results to an HTML file, click “Report.” The HTML file is often easier to read than the Results dialog.

- To save results to a text file, click “Save.” This is useful to:
  - Reload the results later if you are not using the DLN database.
  - View the results in a spreadsheet. The results are tab-delimited and have a header row.
  - Send the results to a colleague to view in their copy of the DLN:Inspector.

### 3.3.2 Loading inspection results

To load inspection results, click “Results” in the main window.

- If you are not using the DLN database, the DLN:Inspector prompts you for the text file containing the results.

- If you are using the DLN database, the DLN:Inspector displays the following dialog:

  ![Image of DLN:Inspector](image)

  Select the project, group, image set and the date/time of the inspection, then click “Results.”
3.3.3 Understanding inspection results

This section discusses inspection results.

3.3.3.1 Why are there so many results?

The first question many people ask when they see their inspection results is, “Why are there so many results?” To answer this question, you need to understand how the DLN:Inspector works.

Suppose you have image files image1.dng through image10.dng and image1.jpg through image10.jpg. For each property, the DLN:Inspector checks:

- The value in image1.dng against the values in image2.dng through image10.dng.
- The value in image1.jpg against the values in image2.jpg through image10.jpg.
- The value in image1.jpg against the value in image1.dng.

Now suppose that a property value in image7.jpg and image7.dng is different from the value in the other files. This results in two error results — one when comparing image1.dng against image7.dng and one when comparing image1.jpg against image7.jpg.

But what happens if the value in image1.jpg and image1.dng is different from all the other files? This results in 18 error results — nine when comparing image1.dng against the other DNG files and nine when comparing image1.jpg against the other JPEG files.

The DLN:Inspector checks more than 150 properties for each file, so a huge number of results is possible.

3.3.3.2 What do the results mean?

There are four broad categories of results:

- **OK.** No problems found.
- **Warning.** Something you need to investigate further. It may or may not be a problem.
- **Error.** A problem that occurred during image capture or processing. This should be fixed before you generate an RTI file or photogrammetry model.
- **Fatal error.** Programming or configuration errors. You should never encounter these.

The most common errors and warnings are:

- **Does not satisfy restrictions.** Some properties are restricted to a certain value, set of values, or range of values. For example, the `crs:SharpenDetail` property must be set to 0. That is, no sharpening is allowed. Not meeting restrictions is sometimes a warning and sometimes an error.
• **Different property values.** The values in two different images are different. For example, `exif:BodySerialNumber` is 123-456-78 in one photo and 321-654-87 in another photo. Different property values is always an error.

• **Different null values.** A null value is a value that is missing. This error is returned if a property has a value in one image and does not have a value in another image. This is always an error. (It is OK if a value is null in all images. In fact, this is common, as many cameras set only some properties.)

For a complete list of error messages and what they mean, see Appendix B, “Results.”

### 3.3.3.3 What does this property mean?

The meaning of many properties can be understood from their names. For example, `exif:LensSerialNumber` is the serial number of the lens used to capture the image.

Other properties are more difficult to understand. For example, `exif:ExposureMode` has three legal values: 0 for auto, 1 for manual, and 2 for auto bracket.

And some properties are almost impossible to decipher, either because they are poorly described or they are not described in any international standard. An example of the latter is most transformation (Xform) properties, which relate to fields in Adobe Camera Raw.

For more information, see the list of international standards in Appendix C, “Properties.”

### 3.4 Setting options

The Options button on the main window displays the following dialog:
The following options are available:

- **Use the DLN.** Whether the DLN:Inspector uses the DLN database. You must restart the DLN:Inspector for this option to take effect.

- **Log all tests to a file.** The DLN:Inspector logs information about all tests to a .log file in the project or group directory. This is generally used only for debugging purposes; use the DLN database for permanently recording results.

- **Stop on error.** If the DLN:Inspector encounters an error, such as two images using different lenses, it will stop inspecting and display the results of the tests it has performed so far.

- **Write inspection results to DLN.** When to write results to the DLN database. This is useful if you don’t want to record results of failed inspections.

- **Remove previous inspection results from DLN.** Whether to remove previous inspection results from the DLN database. This is useful if you inspect the same image sets multiple times, fix errors when you find them, and only record the final results.

- **Maximum hours between two date-times still considered equal.** It is not possible to exactly compare the dates and times that images were captured; doing so would always result in errors. This field specifies the maximum difference between two date-times that are considered to be equal. The default is 1 hour and you can use up to 49 hours.

Note that the options for the RTI and Photogrammetry inspectors are stored separately.
Appendix A: Installation and startup

A.1 Updating the DLN database

If you have installed the DLN: Capture Context (DLNCC) tool, you should update the DLN database before using the DLN:Inspector. If you have not installed the DLNCC tool or you do not want the DLN:Inspector to use the DLN database, you can skip this step.

A.1.1 On MacOS

Note that in future versions of the software, this process will not be needed, as the database will already have the needed updates for Inspector. In other words, this step is only required in the Beta version of the software.

To update the DLN database on MacOS:

1. If you installed DLNCC using its installer and in the standard way, it will have installed the Postgres Database in the listed directory, and set up a default account with the user name “postgres” and you should not need to find the directory (step 2) or edit the file (step 4 – 5). You will need to perform steps 3 and 6 – 8.

2. Find the path of the folder containing the PostgreSQL psql utility. This is in a bin folder beneath the PostgreSQL installation directory. A common path for this utility is:

   /Applications/Postgres.app/Contents/Versions/latest/bin

   Note that to look at the contents of an application, right click in the Finder and choose “Show Contents”

3. Find the path of the MacOS subfolder for either the Inspector-PG or Inspector-RTI applications. (note that you only need to find and run the script for one of the versions of the tool) Generally this is:

   /Applications/DLN-Inspector-PG.app/Contents/MacOS

4. Using a text editor, open the updatepg.sh shell script in the scripts subfolder. The script contains a single line.

   WARNING: Do not double-click on the updatepg.sh file. This may run the script. You need to edit it.

5. Check that the path of the psql utility and the user name are correct. The path is everything from the start of the line up to psql. The user name is the word after the -U.

   For example, the following line uses the path from step 1 and the user name postgres, which is the default:
/Applications/Postgres.app/Contents/Versions/latest/bin/psql
- U postgres -d cpt_db -f $1

6. Open the Terminal. (Open the Applications folder, open the Utilities folder, and double-click on the Terminal application.)

7. Use the cd command to change to the scripts subfolder. For example, to go to the folder in step 2, type the command:

    cd /Applications/DLN-Inspector-PG.app/Contents/MacOS

8. Type the following commands to update the database:

    chmod 755 updatepg.sh

    ./updatepg.sh createpg.sql

---

A.1.2 On Windows

To update the DLN database on Windows:

1. Find the path of the directory containing the PostgreSQL psql utility. This is in a bin directory beneath the PostgreSQL installation directory. A common path for this utility is:

    C:\Program Files\PostgreSQL\<version-number>\bin

2. Find the path of the scripts subfolder of the DLN:Inspector installation folder. By default, this is:

    C:\Program Files (x86)\CHI\DLN-Inspector\scripts

3. Using a text editor such as Notepad, open the updatepg.cmd shell script in the scripts subfolder. The script contains a single line.

    WARNING: Do not double-click on the updatepg.cmd file. This will run the script. You need to edit it.

4. Check that the path of the psql utility and the user name are correct. The path is everything from the start of the line up to psql. The user name is the word after the -U.

    For example, the following line uses the path from step 1 and the user name postgres, which is the default:

    c:\"program files"\postgresql\9.5\bin\psql -U postgres
    -d cpt_db -f $1
NOTE: If a directory in your path has spaces in it, you must surround that directory name with double quotes.

5. Open the command line. (Click Start, type cmd.exe in the Search box, and click on cmd.exe.)

6. Use the cd command to change to the scripts subfolder. For example, to go to the folder in step 2, type the command:

   C:"Program Files (x86)"\CHI\DLN-Inspector\scripts

   NOTE: If a directory in your path has spaces in it, you must surround that directory name with double quotes.

7. Type the following command to update the database:

   updatepg.cmd createpg.sql

A.2 Solving startup problems

If you have problems starting the DLN:Inspector for the first time, see the following sections.

A.2.1 Database installed but not running (or other connection problem)

You will see a dialog similar to this:

![Error dialog](image)

The error message explains why the DLN:Inspector cannot connect to the database. The most common problem is that the database is not running.

If you want to fix the connection problem:

2. Fix the connection problem, such as by starting the database.
3. Restart the DLN:Inspector.

If you want to continue without the database:

1. Click “Yes.”

2. The DLN:Inspector asks if you want to use the database in the future. Click “No.” (You can change your answer later. For more information, see section 3.4, “Setting Options.”) The DLN:Inspector continues without the database.

**A.2.2 Database installed and running but not updated**

If the DLN database is installed and running but not updated for the DLN:Inspector, you will see the following dialog box:

![Database not updated dialog box]

If you want to update the database:


2. Update the database. For more information, see section A.1, “Updating the DLN database.”

3. Restart the DLN:Inspector.

If you want to continue without the database:

1. Click “Yes.”

2. The DLN:Inspector asks if you want to use the DLN database in the future. Click “No.” (You can change your answer later. For more information, see section 3.4, “Setting Options.”) The DLN:Inspector continues without the database.

**A.3 Creating custom directory patterns (advanced)**

The custom directory patterns are defined in two files, DirPatterns-RTI.json and DirPatterns-Photo.json. These files use the JSON data interchange format. JSON is used to pass data between programs, especially programs that run on the Web.
A.3.1 JSON syntax

JSON is built on a small number of parts. The parts you need to know are:

- **Strings.** A series of characters, such as “Yes”. Strings are different from numbers.

- **Property-value pairs:** A property name, a colon, and a property value, which is a string, an array, or an object.

- **Objects.** A list of property-value pairs in which the order does not matter.

- **Arrays.** A list of strings or objects in which the order does matter.

A string is written with double quotes ("). For example, this is the string “Yes” in JSON:

"Yes"

Spaces inside strings matter. For example, these strings are different ("⋅" represents a space):

"Yes"
"⋅⋅⋅Yes⋅⋅⋅"

Spaces outside strings don’t matter. For example, these strings are the same:

"Yes"
"Yes"

The case of letters in a string matters. These are different strings:

"Yes"
"yes"

A property-value pair is a property name (a string), a colon, and a property value (another string). For example, the following means that the “HasSetName” property has the value “No”.

"HasSetName" : "No"

An object is a set of braces ({}) wrapped around one or more property-value pairs, separated by commas. For example, the following object has three properties: Contains, HasSetName, and HasTypeName:

```
{
  "Contains" : "Dirs",
  "HasSetName" : "No",
  "HasTypeName" : "No"
}
```

Remember, spaces outside double quotes don’t mean anything, so this is the same object:
"Contains":"Dirs","HasSetName":"No","HasTypeName":"No"}

**IMPORTANT:** There are commas after all property-value pairs except the last.

An array is a list of values (strings or objects) surrounded by square brackets ([]). For example, this is an array of strings:

```
[  
  "Museum of History Project",
  "Cup",
  "original-captures",
  "image-1.dng",
  "image-2.dng",
  "...
]
```

And this is an array of objects:

```
[  
  {  
    "Contains" : "Dirs",
    "HasSetName" : "No",
    "HasTypeName" : "No"
  },  
  {  
    "Contains" : "Dirs",
    "HasSetName" : "Yes",
    "HasTypeName" : "No"
  }
]
```

Again, notice that there are commas after all strings in the first array except the last, and after all objects in the second array except the last.

**A.3.2 Directory pattern files**

Open DirPatterns-Photo.json and look at it, then read the following description.

**WARNING: DO NOT** make any changes to this file!

The file consists of a single object, which is represented by the outermost braces ({}). This object has a single property, DirPatterns. The value of this property is an array of objects. (The array is surrounded by square brackets [[]] and each object is surrounded by braces ({}). Each object represents a single directory pattern.)

In plain English, the directory pattern file contains a list of directory patterns. The DLN:Inspector reads the list of directory patterns and displays them in the order they appear in the directory patterns file.
The directory pattern file consists of the following parts:

- **Summary property.** The value of the property is what will be displayed after the radio button in the “Directory patterns” area in the main window of the DLN:Inspector. (See section 3.2.3, “DLN:Inspector for Photogrammetry, without the DLN database (browsing)” for a picture of this area.)

  **Note:** The DLN:Inspector for RTI only displays this area if there are at least two patterns.

- **Example property.** This is an array of strings that is displayed when you click on the example link in the “Directory patterns” area. Each string is displayed on a separate line.

- **Patterns property.** This is an array of objects, each of which represents a single directory pattern. The order of the objects is important: The first object is the pattern for the top-most directory (the directory selected with the Browse button), the second object is the pattern for its subdirectories, the third object is the pattern for the sub-subdirectories, and so on.

Each directory pattern has the following properties:

- **Contains property.** Whether a directory contains subdirectories (“Dirs”) or image files (“Files”).

- **HasSetName property:** Whether a directory name should be used as an image set name (“Yes” or “No”).

- **HasTypeName property:** Whether a directory name contains a type name (“Yes” or “No”).

- **TypeRegexes property:** An array of objects used to determine an image type, such as JPEG, from a directory name. If HasTypeName is “Yes”, this property must be present. If HasTypeName is “No”, it must not be present.

Each object in the TypeRegexes array has the following properties:

- **Type property.** The name of the image type (“JPEG”, “TIFF”, or “DNG”).

- **Regex property.** A regular expression that provides a pattern to match.

### A.3.2.1 Regular Expressions

Regular expressions are a pattern-matching language. The purpose of the objects in the TypeRegexes array is to match an image type with a directory name. For example, the following object says that when a directory name contains the (case-independent) string “TIF” in it (the Regex property), the directory contains TIFF files (Type property):

```json
{
   "Type" : "TIFF",
   "Regex" : "TIF"
}
```
Note that this regular expression will match any directory name with “TIF” in it, including “TIF files”, “Tiffany”, and “rotifera”. You will generally have at least two objects in the TypeRegexes array, one to match the name of the directory containing archived (DNG) images and one to match the name of the directory containing processed images.

If you want to be more specific, you can match exact names. For example, the following regular expression exactly matches the string “TIF files”:

"^TIF files$"

The caret (^) at the start of the string says the directory name must start with the next character — in this case, “T” — and the dollar sign ($) at the end of the string says that the string must end with the previous character — in this case, “s”.

You can also match one or more strings. For example, the following regular expression says that the directory name must contain “TIF” or “TIFF” to match. (The pipe (|) symbol means OR.)

"TIF|TIFF"

For a complete description of the rules used by regular expressions, see the “Detailed Description” section of QRegexp.

A.3.2.1 Describing directory structure

The objects in the Patterns property describe directory structure. Each object describes a single directory level. For example, suppose you had the following directory structure:

```
project-directory
   |
image-set-directory
   /          \
Archival      Images
```

You would use this Patterns array:

"Patterns" :
[
  {
    "Contains" : "Dirs",
    "HasSetName" : "No",
    "HasTypeName" : "No"
  },
  {
    "Contains" : "Dirs",
    "HasSetName" : "Yes",
    "HasTypeName" : "No"
  },
  {
    "Contains" : "Dirs",
    "HasSetName" : "Yes",
    "HasTypeName" : "No"
  },
  {
    "Contains" : "Dirs",
    "HasSetName" : "Yes",
    "HasTypeName" : "No"
  }
]
"Contains" : "Files",
"HasSetName" : "No",
"HasTypeName" : "Yes",
"TypeRegexes" :
[ 
  {"Type" : "JPEG", "Regex" : "^Images$"},
  {"Type" : "DNG", "Regex" : "^Archival$"} 
] 
] 

The first object matches the top-level directory, which is the project directory. The project directory contains subdirectories ("Contains": "Dirs"), its name is not used as the image set name ("HasSetName": "No"), and its name does not contain type information ("HasTypeName": "No").

The second object matches next level of directories (image sets). Each directory contains subdirectories, its name is used as the image set name, and its name does not contain type information. Note that this pattern can match multiple directories at this level.

The third object matches the bottom level directories. Each directory contains files, its name is not used as the image set name, and its name does contain type information. The objects in the TypeRegexes array use very simple regular expressions. These state that if the directory name is “Images”, the directory contains JPEG files, and if it is “Archival”, the directory contains DNG files. Directories at this level with other names are ignored.

A.3.3 Creating a directory pattern file

To create a new directory pattern file:

1. Copy of one of the existing directory pattern files (DirPatterns-RTI.json or DirPatterns-Photo.json).

2. Copy an existing pattern
d{"Summary":...,"Example":...,"Patterns":[]}and paste it after the last pattern. Be sure to put a comma between the next-to-last pattern and the pattern you just pasted.

3. Change the value of the Summary property. This is what will be displayed as the label of the radio button for this pattern.

4. Change the value of the Example property. This is what will be displayed when you click on the example link.

5. Create one pattern object (element in the Patterns array) for each directory you want to match.
• The order of the pattern objects must match the hierarchical structure of your directories. The top-most pattern will match the directory selected with the Browse button.

• Exactly one pattern object must have a HasSetName property with a value of “Yes”.

• Exactly one pattern object must have a HasTypeName property with a value of “Yes”. This object must also have a TypeRegexes property.

• The last pattern object must have a Contains property with a value of “Files”. The other objects must have a Contains property with a value of “Dirs”.

• Commas must separate the pattern objects, as well as the objects in the TypeRegexes array.

6. Remove any old patterns that you do not want to use.

   Remember that when using the DLN:Inspector for RTI, DirPatterns-RTI.json must contain at least two patterns or the DLN:Inspector will not display the patterns. This restriction does not apply to the DLN:Inspector for Photogrammetry.

7. Check that you have not made any JSON syntax errors by using an online JSON viewer, such as JSON Viewer. Do not continue until your file can be viewed without errors.

8. Rename the existing pattern file to use an .original suffix. For example, change the name of DirPatterns-RTI.json to DirPatterns-RTI.json.original.

9. Rename your pattern file to DirPatterns-RTI.json or DirPatterns-RTI.json, as appropriate.

10. Run DLN:Inspector to test your pattern file. DLN:Inspector will detect many, but not all, problems with pattern files.
Appendix B: Results

The following is a list of results that can be returned by the DLN:Inspector. Results have a short description and a full description. The full description gives a detailed description of the problem and is specific to the context in which the error occurred. For example, it might list the files and properties involved with the error.

The following tables list short descriptions. In place of the full description, they give a general explanation of the problem and what you should do to fix it.

If you think you have received an error incorrectly, please contact CHI. There appear to be variations in how different manufacturers record metadata and CHI would like to be able to document these differences.

B.1 OK results

These are results that say a section of the inspection completed successfully.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Capture tests OK</td>
<td>No image capture errors were found in this capture set.</td>
</tr>
<tr>
<td>DLN updated</td>
<td>Information about the inspection was successfully added to the DLN database</td>
</tr>
<tr>
<td>File tests OK</td>
<td>No file errors were found in this capture set.</td>
</tr>
<tr>
<td>Transformation tests OK</td>
<td>No transformation errors were found in this capture set.</td>
</tr>
</tbody>
</table>

B.2 Warnings

Warnings indicate a possible problem that the user needs to investigate further. In some cases the situation is acceptable; in other cases it is not. For example, a DNG file might exist without a corresponding JPEG or TIFF file. If the DNG file is a color card, this is OK; if it is part of the image set, this is an error.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Invalid rational number</td>
<td>The values of some properties, such as aperture value, are stored as rational numbers. This warning indicates that a rational number has an invalid value, such as when the denominator is 0.</td>
</tr>
</tbody>
</table>
whether it makes sense for the property to have an invalid rational number as a value. You then need to decide whether this is an error or can be safely ignored.

<table>
<thead>
<tr>
<th><strong>Missing JPEG/TIFF file</strong></th>
<th>No JPEG/TIFF file was found corresponding to a particular DNG file. You must check whether this is acceptable. For example, if the DNG file is a color test, it is acceptable for there to be no corresponding JPEG/TIFF file. On the other hand, if no JPEG/TIFF file is found corresponding to a DNG image of the object being studied, then this is an error and a JPEG/TIFF file needs to be generated from the DNG file.</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Multiple DLN values</strong></td>
<td>A query for a property stored in the DLN database returned multiple values. This may make sense in some cases, but is not supported by the DLN:Inspector.</td>
</tr>
<tr>
<td><strong>No DNG files</strong></td>
<td>No DNG files were found in the archived images directory. If no DNG files were created — for example, images were captured as JPEGs — this is acceptable. If DNG files were created, then they are not in the archived images directory and should be moved there.</td>
</tr>
<tr>
<td><strong>No metadata found</strong></td>
<td>No metadata was found in an image file. While there is nothing strictly wrong with this, it is a surprising result and may indicate problems. The most likely cause is that a tool used to process the image stripped the metadata out of it. As a result, you cannot inspect the image file with the DLN:Inspector or retrieve the image metadata at a later point in time.</td>
</tr>
<tr>
<td><strong>Not using DLN</strong></td>
<td>The DLN database was not found / used. The DLN:Inspector can operate without the DLN database, but cannot compare the camera identity tags in the DLN database against the same tags in the image files. Note that even without the DLN database, these tags are still compared among the image files.</td>
</tr>
</tbody>
</table>
B.3 File errors

File errors are errors having to do with missing files.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Missing DNG file</td>
<td>No DNG file was found corresponding to a particular JPEG/TIFF file. If there are any DNG files, then all JPEG/TIFF files must have corresponding DNG files. If a DNG file is missing, it could be because it was accidentally deleted or because a JPEG/TIFF file unrelated to RTI or photogrammetry was accidentally placed in the processed images directory. You will need to find the cause of this error and fix it.</td>
</tr>
</tbody>
</table>

B.4 General errors

General errors can occur during image capture or image transformation.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different array sizes</td>
<td>Some properties have multiple values; that is, they are an array. For example, the values of crs:ToneCurvePV2012 give the (x, y) coordinates of the tone curve. This error means that the same property has different numbers of values in different images. For example, this might occur if different tone curves were applied to each image. This error generally means that different transformations were applied to each image. You will need to investigate to be sure.</td>
</tr>
</tbody>
</table>
| Different null values   | A property has a value in one image and does not have a value in another image (or the DLN database). There are a variety of reasons for this problem. For example:  
• The value is set in the DLN database and not in an image file because the camera did not record it in the image file.  
• The value is set in the DNG file and not in the JPEG file because the JPEG file was created by Adobe Camera Raw. (It appears that Adobe Camera Raw does not pass all known metadata values from DNG files to JPEG files.)  
• The value is not set in an image file for unknown reasons. You should carefully trace the cause of this problem. |
| Different property values | The property has different values in two different image files (or in an image file and the DLN database). There is no single cause of this error. For example, it could be |
due to changing camera settings while capturing images or to applying a transformation in Adobe Camera Raw to one image but not the other image.

If one of the properties is in the DLN database, it could also be due to entering an incorrect value in the DLNCC tool. This is especially likely with things like camera and lens names. You will need to investigate the cause of this error carefully.

| Does not satisfy restrictions | The property did not have a valid value. Many properties have restrictions on what values they can have. For example, exif:ExposureProgram should have a value of 1, indicating that the camera was set to manual exposure mode, as opposed to aperture priority or shutter priority. And crs:Contrast should have a value of 0, indicating that the contrast was not changed in Adobe Camera Raw. If this error occurs, you will generally need to fix the problem, such as by removing contrast adjustments in a DNG file and regenerating the corresponding JPEG files. It appears that in some cases cameras might not record metadata properly, leading to this error. You can test this by setting the camera correctly, taking a photo, processing it, and then testing if this error occurs. If it does, please contact CHI so they can document the problem. |

**B.5 Capture errors**

Capture errors are errors that occur during image capture.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date difference too big</td>
<td>The value of the exif:DateTimeOriginal property will differ across the images in a capture set because no two images can be captured at the same time. Because of this, date/times cannot be compared exactly. Instead, two date/times must be within a certain number of hours of each other to be considered equal. You can specify this number in the Options dialog. If the specified number of hours is too short — for example, it is set to 1 and your capture session took two hours — change the number of hours and re-run the inspection. If the specified number of hours is reasonable, you will need to investigate the cause of this error. One possible cause is that the incorrect date/time was entered in</td>
</tr>
</tbody>
</table>
the DLNCC tool. Another possible cause is that metadata date/times do not specify the time zone, so all time zones are assumed to be the same. If images were shot in one time zone and processed in another time zone, this error may occur.

Different synonym values

Some properties are synonymous. For example, \texttt{aux:Lens}, \texttt{exif:Lens}, and \texttt{exifEX:Lens} all appear to mean the same thing.

It is expected that a processing chain (camera and set of processing tools) will only use one synonym. If this error occurs, you will need to find out where the additional synonyms were introduced. Also, please contact CHI and send them the full description of this error.

DLN database missing data

No data was found in the DLN database for a property. Either ignore the error or enter the value with the DLNCC tool and reinspect the images.

Invalid DLN value

The format of a property in the DLN database is different from the expected format.

Please contact CHI and send them the full description of the error.

Required DLN value is null

A required property in the DLN database was null (missing).

To fix this error, enter the correct value using the DLNCC tool.

Y dimension is null

The Y dimension of an image is null (missing) and the X dimension is non-null (present).

The cause of this error is not clear, but it probably indicates a problem during image processing and may require the images to be regenerated or recaptured.

### B.6 Transformation errors

Transformation errors are errors that occur during image transformation, such as with Adobe Camera Raw.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Different dimensions</td>
<td>Two images have different dimensions.</td>
</tr>
<tr>
<td></td>
<td>Images may be rotated or resized during processing. However, all images of a given type (JPEG, TIFF, or DNG) must have the same height and width.</td>
</tr>
<tr>
<td></td>
<td>JPEG/TIFF images may have a different width and height than DNG images. However, the aspect ratio (ratio of width to height) of the JPEG/TIFF images must be the same as or the</td>
</tr>
</tbody>
</table>
inverse of the aspect ratio of the DNG images. That is, the size of the JPEG/TIFF images must be obtained by rotating and/or resizing the DNG images and not by cropping.

If this error occurs for two images of the same type (DNG, JPEG, or TIFF), then all images of that type must be regenerated so they have the same width and height. If this is not possible, then the images must be recaptured — that is, the photos must be retaken.

If this error occurs for two images of different types, then the JPEG/TIFF images must be regenerated from the DNG images in such a way that all JPEG/TIFF images have the same width and height.

### B.7 Fatal errors

Most fatal errors should never be encountered; they represent programming or configuration errors caught by the DLN:Inspector. Users can encounter a few fatal errors, such as there not being any JPEG/TIFF files.

<table>
<thead>
<tr>
<th>Short Description</th>
<th>Explanation</th>
</tr>
</thead>
</table>
| Configuration error | The PropsDLN-Xxx.xml configuration file is missing an SQL query that specified how to retrieve a property from the DLN database. The full description lists the property that is missing the query.  
There are two ways to fix this. First, uninstall and reinstall the DLN:Inspector. Second, if you downloaded one of these files after installing the DLN:Inspector, download it again. If neither solution works, contact CHI and send them the information in the long description of the error. |
| Different property types | A property has different data types in different images. For example, its data type in one image was integer and in another type was rational. Please contact CHI. |
| Different structure sizes | Some properties, such as `exif:Flash`, are structures containing other properties. This error occurs when the size of the structure is different for the same property in two different images. Please contact CHI. |
| Error querying DLN database | An SQL query used to retrieve a property from the DLN database returned an error.  
This could be due to a bug in the query, because the format of the database has changed, or because the property was missing from the database. |
<table>
<thead>
<tr>
<th>Error writing file</th>
<th>Please contact CHI and send them the full description of the error.</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fatal Error</td>
<td>An error occurred while writing to the log file.</td>
</tr>
<tr>
<td>Invalid data type</td>
<td>A property was included in the list of internal properties that have synonyms, but no code was found to process it.</td>
</tr>
<tr>
<td>No JPEG/TIFF files</td>
<td>The data type of the property was unknown. This is probably due to a configuration error in DLN:Inspector. Please contact CHI and send them the full description of the error.</td>
</tr>
<tr>
<td>No JPEG/TIFF files found</td>
<td>No JPEG/TIFF files were found in the processed images directory. This error occurs if the JPEG/TIFF files were not created or if they were accidentally placed in the wrong directory. Create and/or move them to the processed images directory.</td>
</tr>
<tr>
<td>No property information found</td>
<td>No information was found for a property that is related to the property being inspected. This occurs because the configuration information for the related property was missing. Please contact CHI.</td>
</tr>
<tr>
<td>No value found</td>
<td>No value was found for a property that is related to the property being inspected. This usually occurs because the configuration information for the related property was missing. Please contact CHI.</td>
</tr>
<tr>
<td>Programming error</td>
<td>There is a programming error in the DLN:Inspector. Please contact CHI and send them the full description of the error.</td>
</tr>
<tr>
<td>Synonyms must have same type</td>
<td>Some properties are synonymous. For example, aux:Lens, exif:Lens, and exifEX:Lens all appear to mean the same thing. Synonymous properties must have the same type. Please contact CHI and send them the full description of this error.</td>
</tr>
<tr>
<td>XMP SDK error</td>
<td>The Adobe XMP SDK, which is used to retrieve metadata from image files, returned an error. Please contact CHI and send them the full description of the error and the image file that caused the error.</td>
</tr>
<tr>
<td>XMP SDK exception</td>
<td>The Adobe XMP SDK, which is used to retrieve metadata from image files, returned an unknown error. Please contact CHI and send them the full description of the error and the image file that caused the error.</td>
</tr>
</tbody>
</table>
Appendix C: Properties

Image metadata is mostly based on several international standards. Unfortunately, different manufacturers implement different parts of these standards, as well as using metadata not found in any standard. Transformation (Xform) properties generally refer to transformations performed in Adobe Camera Raw.

C.1 Property files

This manual does not list the properties inspected by the DLN:Inspector. You can find a list of these properties in the PropsXxxxx-xxx.xml files in the installation directory, where Xxxxx is the name of the class of properties (CameraID, Capture, DLN, or Xform) and Xxx is RTI or Photo.

You can open these files with a text editor and find the properties in the <Namespace> and <Name> tags. For example, the following entry in PropsCapture-RTI.xml describes the exif:ColorSpace property:

```
<PropInfo>
  <Namespace>http://ns.adobe.com/exif/1.0/</Namespace>
  <Name>ColorSpace</Name>
  <IsArray>false</IsArray>
  <SimpleType>
    <Type>Integer</Type>
  </SimpleType>
</PropInfo>
```

Some properties have been commented out because they were found to be set inconsistently, not represent what they seemed to represent, or did not apply. These properties are surrounded by <!-- and -->. For example, the exif:ColorSpace property is commented out in the PropsCapture-Photo.xml file:

```
<!--
<PropInfo>
  <Namespace>http://ns.adobe.com/exif/1.0/</Namespace>
  <Name>ColorSpace</Name>
  <IsArray>false</IsArray>
  <SimpleType>
    <Type>Integer</Type>
  </SimpleType>
</PropInfo>
-->
```

The DLN:Inspector does not check commented out properties. They remain in the property files as a convenience, as they may be used in the future. If you are curious about what the rest of the information in this file means, see the comments in PropInfo.dtd. DO NOT edit any of these files!
C.2 Property prefixes

You may have noticed that properties in the results have prefix and a name:

```plaintext
exif:ColorSpace
```

while properties in the property files have a URL and a name:

```xml
<Namespace>http://ns.adobe.com/exif/1.0/</Namespace>
<Name>ColorSpace</Name>
```

The reason for this comes from Namespaces in XML, of which you only need to understand the following:

- Properties have a two-part name, a URL (namespace) and a property name. For example, `http://ns.adobe.com/exif/1.0` and `ColorSpace`.

- URLs group related property names. For example, `http://ns.adobe.com/exif/1.0` groups all the properties in version 1.0 of the Exif standard.

- URLs are chosen because they are unique. They do not point to anything. If you put them in a Web browser, it is unlikely you will find anything.

- URLs are commonly replaced by prefixes to make property names more readable. For example, `exif:ColorSpace` means the `ColorSpace` property in the `http://ns.adobe.com/exif/1.0` namespace.

The following table maps prefixes (which are used in results and in the standards listed in appendix C.3, “International standards”) to URLs (which are used in the property files):

<table>
<thead>
<tr>
<th>Specification</th>
<th>Prefix</th>
<th>Namespace</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exif 2.2 or earlier</td>
<td>exif</td>
<td><a href="http://ns.adobe.com/exif/1.0/">http://ns.adobe.com/exif/1.0/</a></td>
</tr>
<tr>
<td>Exif 2.21 or later</td>
<td>exifEX</td>
<td><a href="http://cipa.jp/exif/1.0/">http://cipa.jp/exif/1.0/</a></td>
</tr>
<tr>
<td>Exif 2.3 *</td>
<td>aux</td>
<td><a href="http://ns.adobe.com/exif/1.0/aux/">http://ns.adobe.com/exif/1.0/aux/</a></td>
</tr>
<tr>
<td>TIFF 6.0</td>
<td>tiff</td>
<td><a href="http://ns.adobe.com/tiff/1.0/">http://ns.adobe.com/tiff/1.0/</a></td>
</tr>
<tr>
<td>XMP</td>
<td>xmp</td>
<td><a href="http://ns.adobe.com/xap/1.0/">http://ns.adobe.com/xap/1.0/</a></td>
</tr>
<tr>
<td>Dublin Core</td>
<td>dc</td>
<td><a href="http://purl.org/dc/elements/1.1/">http://purl.org/dc/elements/1.1/</a></td>
</tr>
<tr>
<td>Adobe Camera Raw</td>
<td>crs</td>
<td><a href="http://ns.adobe.com/camera-raw-settings/1.0/">http://ns.adobe.com/camera-raw-settings/1.0/</a></td>
</tr>
</tbody>
</table>

* These properties were included in earlier versions of the XMP specification, but removed in 2012 when “Exif 2.3 Metadata for XMP” (which introduced the exifEX prefix) was released. The DLN:Inspector uses them because they are still found in image files.
C.3 International standards

The following standards describe individual properties.

NOTE: The transformation properties, which have the prefix crs, are not documented anywhere. They apparently correspond to settings in Adobe Camera Raw (ACR).

NOTE: Some links may be broken. If a link doesn’t work, try entering it in the Wayback Machine (http://web.archive.org/).

### Exif

<table>
<thead>
<tr>
<th>Specification</th>
<th>URL</th>
<th>Official Website?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exif 2.2</td>
<td><a href="http://web.archive.org/web/20131019050323/http://www.exif.org/Exif2-2.PDF">http://web.archive.org/web/20131019050323/http://www.exif.org/Exif2-2.PDF</a></td>
<td>N</td>
</tr>
<tr>
<td>Exif 2.21 (Draft)</td>
<td><a href="http://web.archive.org/web/20051228234707/http://tse.jeita.or.jp/avs/data/cp3451_1.pdf">http://web.archive.org/web/20051228234707/http://tse.jeita.or.jp/avs/data/cp3451_1.pdf</a></td>
<td>N</td>
</tr>
<tr>
<td>Exif 2.3</td>
<td><a href="http://www.cipa.jp/std/documents/e/DC-008-2012_E_C.pdf">http://www.cipa.jp/std/documents/e/DC-008-2012_E_C.pdf</a></td>
<td>Y</td>
</tr>
<tr>
<td>Exif 2.3 Metadata for XMP</td>
<td><a href="http://www.cipa.jp/std/documents/e/DC-010-2012_E.pdf">http://www.cipa.jp/std/documents/e/DC-010-2012_E.pdf</a></td>
<td>Y</td>
</tr>
</tbody>
</table>

### TIFF

<table>
<thead>
<tr>
<th>Specification</th>
<th>URL</th>
<th>Official Website?</th>
</tr>
</thead>
<tbody>
<tr>
<td>TIFF 6.0</td>
<td><a href="https://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf">https://www.itu.int/itudoc/itu-t/com16/tiff-fx/docs/tiff6.pdf</a></td>
<td>Y</td>
</tr>
</tbody>
</table>

### DNG

<table>
<thead>
<tr>
<th>Specification</th>
<th>URL</th>
<th>Official Website?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Specification</td>
<td>URL</td>
<td>Official Website?</td>
</tr>
<tr>
<td>---------------</td>
<td>-----</td>
<td>------------------</td>
</tr>
</tbody>
</table>

### IPTC

<table>
<thead>
<tr>
<th>Specification</th>
<th>URL</th>
<th>Official Website?</th>
</tr>
</thead>
<tbody>
<tr>
<td>IPTC</td>
<td><a href="http://www.iptc.org/std/photometadata/specification/IPTC-PhotoMetadata">Link</a></td>
<td>Y</td>
</tr>
</tbody>
</table>

### Dublin Core

<table>
<thead>
<tr>
<th>Specification</th>
<th>URL</th>
<th>Official Website?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dublin Core</td>
<td><a href="http://dublincore.org/documents/dcmi-terms/">Link</a></td>
<td>Y</td>
</tr>
</tbody>
</table>
An additional source of useful information is Phil Harvey’s Exif Tags, which lists the names and legal values of many image metadata properties, including properties not found in any specification. (Phil Harvey is the author of ExifTool, a free, widely-used tool for reading and writing image metadata.)